Downloaded from orbit.dtu.dk on: May 04, 2023

DTU Library

CoreFlow: A computational platform for integration, analysis and modeling of complex biological data

Pasculescu, Adrian; Schoof, Erwin; Creixell, Pau; Zheng, Yong; Olhovsky, Marina; Tian, Ruijun; So, Jonathan; Vanderlaan, Rachel D.; Pawson, Tony; Linding, Rune

Total number of authors:
11

Published in:
Journal of Proteomics

Link to article, DOI:
10.1016/j.jprot.2014.01.023

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Pasculescu, A., Schoof, E., Creixell, P., Zheng, Y., Olhovsky, M., Tian, R., So, J., Vanderlaan, R. D., Pawson, T., Linding, R., & Colwill, K. (2014). CoreFlow: A computational platform for integration, analysis and modeling of complex biological data. *Journal of Proteomics*, *100*, 167-173. https://doi.org/10.1016/j.jprot.2014.01.023

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

JOURNAL OF PROTEOMICS 100 (2014) 167–173

Available online at www.sciencedirect.com

ScienceDirect

www.elsevier.com/locate/jprot

Technical note

CoreFlow: A computational platform for integration, analysis and modeling of complex biological data☆

Adrian Pasculescu[a], *Erwin M. Schoof*[b,1], *Pau Creixell*[b,1], *Yong Zheng*[a], *Marina Olhovsky*[a], *Ruijun Tian*[a], *Jonathan So*[a,c], *Rachel D. Vanderlaan*[a,d,e], *Tony Pawson*[a,d,†], *Rune Linding*[b,*], *Karen Colwill*[a,**]

[a]*Lunenfeld-Tanenbaum Research Institute, Mount Sinai Hospital, Toronto, Ontario, Canada*
[b]*Cellular Signal Integration Group (C-SIG), Center for Biological Sequence Analysis (CBS), Department of Systems Biology, Technical University of Denmark (DTU), Building 301, DK-2800, Lyngby, Denmark*
[c]*Institute of Medical Science, University of Toronto, Toronto, Ontario, Canada*
[d]*Department of Molecular Genetics, University of Toronto, Toronto, Ontario, Canada*
[e]*Department of Cardiac Surgery, University of Toronto, Ontario, Canada*

ARTICLE INFO

Available online 3 February 2014

Keywords:
Computational pipeline
Mass spectrometry
Data analysis
Statistical analysis
Workflow

ABSTRACT

A major challenge in mass spectrometry and other large-scale applications is how to handle, integrate, and model the data that is produced. Given the speed at which technology advances and the need to keep pace with biological experiments, we designed a computational platform, CoreFlow, which provides programmers with a framework to manage data in real-time. It allows users to upload data into a relational database (MySQL), and to create custom scripts in high-level languages such as R, Python, or Perl for processing, correcting and modeling this data. CoreFlow organizes these scripts into project-specific pipelines, tracks interdependencies between related tasks, and enables the generation of summary reports as well as publication-quality images. As a result, the gap between experimental and computational components of a typical large-scale biology project is reduced, decreasing the time between data generation, analysis and manuscript writing. CoreFlow is being released to the scientific community as an open-sourced software package complete with proteomics-specific examples, which include corrections for incomplete isotopic labeling of peptides (SILAC) or arginine-to-proline conversion, and modeling of multiple/selected reaction monitoring (MRM/SRM) results.

Biological significance

CoreFlow was purposely designed as an environment for programmers to rapidly perform

☆ This article is part of a Special Issue entitled: Can Proteomics Fill the Gap Between Genomics and Phenotypes?

* *Correspondence to:* Rune Linding, Cellular Signal Integration Group (C-SIG), Center for Biological Sequence Analysis (CBS), Department of Systems Biology, Technical University of Denmark (DTU), Anker Engelundsvej, Building 301, DK-2800, Lyngby, Denmark. Tel.: +45 2365 1941; fax: +45 4525 2281.

** *Correspondence to:* Karen Colwill, Lunenfeld-Tanenbaum Research Institute, Mount Sinai Hospital, 600 University Avenue, Room 970, Toronto, Ontario, Canada, M5G 1X5. Tel.: +1 416 586 4800x3018; fax: +1 416 586 8869.

E-mail addresses: linding@cbs.dtu.dk (R. Linding), colwill@lunenfeld.ca (K. Colwill).

[1] These authors contributed equally to this publication.

† Deceased.

1874-3919/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jprot.2014.01.023

data analysis. These analyses are assembled into project-specific workflows that are readily shared with biologists to guide the next stages of experimentation. Its simple yet powerful interface provides a structure where scripts can be written and tested virtually simultaneously to shorten the life cycle of code development for a particular task. The scripts are exposed at every step so that a user can quickly see the relationships between the data, the assumptions that have been made, and the manipulations that have been performed. Since the scripts use commonly available programming languages, they can easily be transferred to and from other computational environments for debugging or faster processing. This focus on 'on the fly' analysis sets CoreFlow apart from other workflow applications that require wrapping of scripts into particular formats and development of specific user interfaces. Importantly, current and future releases of data analysis scripts in CoreFlow format will be of widespread benefit to the proteomics community, not only for uptake and use in individual labs, but to enable full scrutiny of all analysis steps, thus increasing experimental reproducibility and decreasing errors.
This article is part of a Special Issue entitled: Can Proteomics Fill the Gap Between Genomics and Phenotypes?

© 2014 Elsevier B.V. All rights reserved.

Mass spectrometry is a computationally intense experimental procedure. A wide variety of software applications have been developed to facilitate analysis, including search engines that interpret the spectra [1–4], databases that hold raw and processed results [5–7], and sophisticated algorithms that calculate statistical significance [8–10]. Even with the availability of all these programs, the ever changing nature of research often requires scientists to develop their own scripts to handle a specific analysis and, to integrate mass spectrometry data with other experimental data. To address this need, we have built a software application, CoreFlow, which provides an organized framework for programmers to rapidly perform, document and share data analysis steps.

CoreFlow is designed to manage all data manipulation steps within a project (Fig. 1A). It is divided into two main sections: database management and analysis (Fig. 1B). The database management section is used for loading, storing and handling of experimental data in a highly efficient yet flexible manner (Supplementary Figs. 1–5). Here, one can perform basic database operations such as creating new tables, modifying existing ones, adding key indexes, and filtering, joining or aggregating data. The analysis section assembles and then tracks computational pipelines using a hierarchical organization of owners, projects, threads and tasks (Supplementary Fig. 6); a thread represents a particular analysis (e.g. Multiple Reaction Monitoring) and, within each thread, a task holds the code for a specific function (e.g., normalization to bait). These pipelines provide a permanent record of all the computational work performed inside each specific project. The standardized format reveals the complete data analysis pipeline to all users, thereby promoting sharing of data, scripts, and results. Within a pipeline, newer tasks are often built using information extracted from previous tasks via application programming interface (API) calls. These calls allow access to a library of data processing scripts for common functions such as data quality checks and normalization, enable re-use of code for data extraction from tables, and provide the ability to obtain content from attachments linked to tasks (Supplementary Fig. 7). To help track task interdependencies, we created a built-in data provenance feature that identifies these calls (Supplementary Fig. 8). Similarly, data within a task is often extracted from multiple database tables, and CoreFlow maps the relationships between the original tables and the resulting temporary and final tables (Supplementary Fig. 9). At task completion, a detailed analysis report can be generated that includes the task description, code, results and data provenance features (Supplementary Text).

New tasks within a thread can easily be added using the blank template provided or by duplicating and editing an existing task. If an existing task is copied, the user only needs to change the parameters of any API calls (in particular, the unique task identifier) and update table names in the SQL scripts. Each task includes administrative features for version tracking, quality control, priority setting, and file attachment (Supplementary Fig. 10). Icons can be added to the task label to serve as visual cues to the task's purpose. Three integrated playgrounds (similar to Google Code Playgrounds) accelerate analysis by allowing for a rapid cycle of script development and testing. The first playground with an embedded Wiki page enables the addition of meta information about the analysis and its purpose, assumptions and caveats (Supplementary Fig. 11). It can also be used to indicate where and how certain parameters can be changed in the code (e.g., cutoff values, number of data points to be included). This is important, as CoreFlow is designed for rapid analysis and programming where developing user-specific interfaces for each task to accommodate parameter entries is not desirable as it would slow the pace of analysis. A second playground allows for complex database queries that pre-process different data types and sources. The final processing occurs in the third playground where users write scripts in various programming languages to handle the data. These scripts can be readily transferred, without any changes, either to a local programming environment for debugging or a high performance system for speed. They may vary in terms of their complexity and level of abstraction, from bioinformatic analysis in Python or Perl to more detailed statistical analysis in R. For each of these programming languages, CoreFlow allows the import and use of their specific libraries and packages as well as creation of new libraries. This permits, for example, large string processing or manipulation and

transformation of sequences (e.g., in silico digestion of proteins into peptides) using BioPython or BioPerl. Libraries in R can also be employed for statistical analysis, data visualization, and immediate rendering of images in SVG, PDF or other graphic formats for inclusion in manuscripts. Results can be exported in compatible formats for visualization or modeling tools such as Cytoscape [11] or DataRail [12].

Due to its flexibility, CoreFlow can be applied to practically any type of data analysis. Our labs have used it in over fifty very diverse projects that include large-scale mass spectrometry, RNAi, next-generation sequencing, imaging and animal models [13–16]. We typically use it in conjunction with complementary tools targeted to a particular experimental application (e.g., MaxQuant [4] and Proteome Discoverer (ThermoScientific) for mass spectrometry quantitation). We quickly move data (in XML, flat or binary file formats) from such applications to the CoreFlow database where we extract relevant data and model it (Figs. 1a, 2, Supplementary Fig. 12).

In a recent paper involving MRM [16], CoreFlow was used to normalize all samples within a time course to the bait, examine reproducibility between samples, and cluster proteins with similar profiles (Fig. 2). It has helped identify systematic errors that are often introduced in experiments and to correct them computationally, leading to cleaner data and improving the accuracy of results. For example, we have estimated and corrected labeling incorporation deficiencies and arginine-to-proline conversions that affect mass spectrometry measurements when using Stable Isotope Labeling by Amino acids in Cell culture (SILAC) (Fig. 3, Supplementary Fig. 13, Supplementary Text).

CoreFlow implements a client–server architecture, supported by the Apache web server on the Linux operating system, with a web interface written in PHP and Perl, and an underlying MySQL database. This implementation provides the advantages of centralized access, resource distribution, back-up capabilities, and easy upgrading. It can be deployed

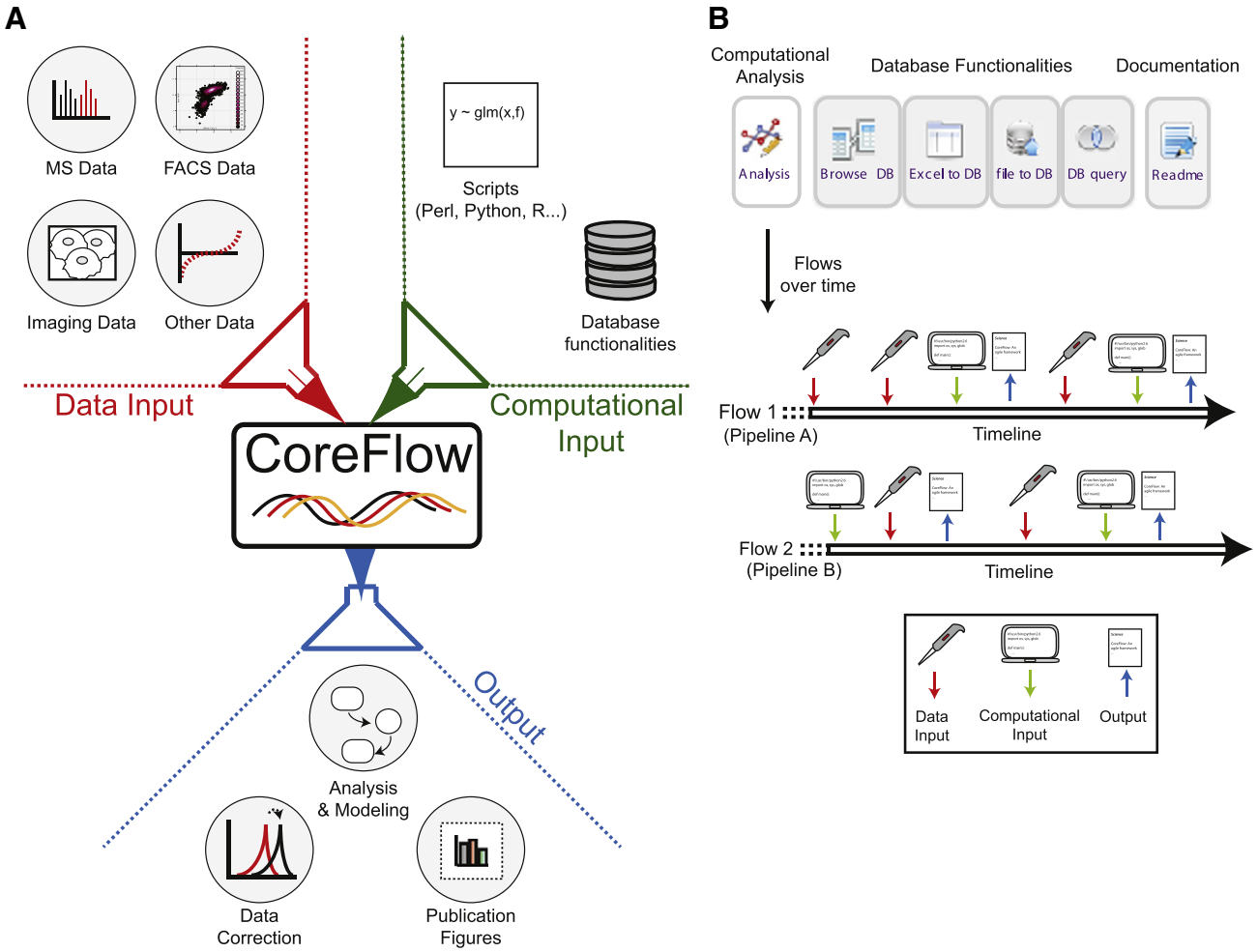
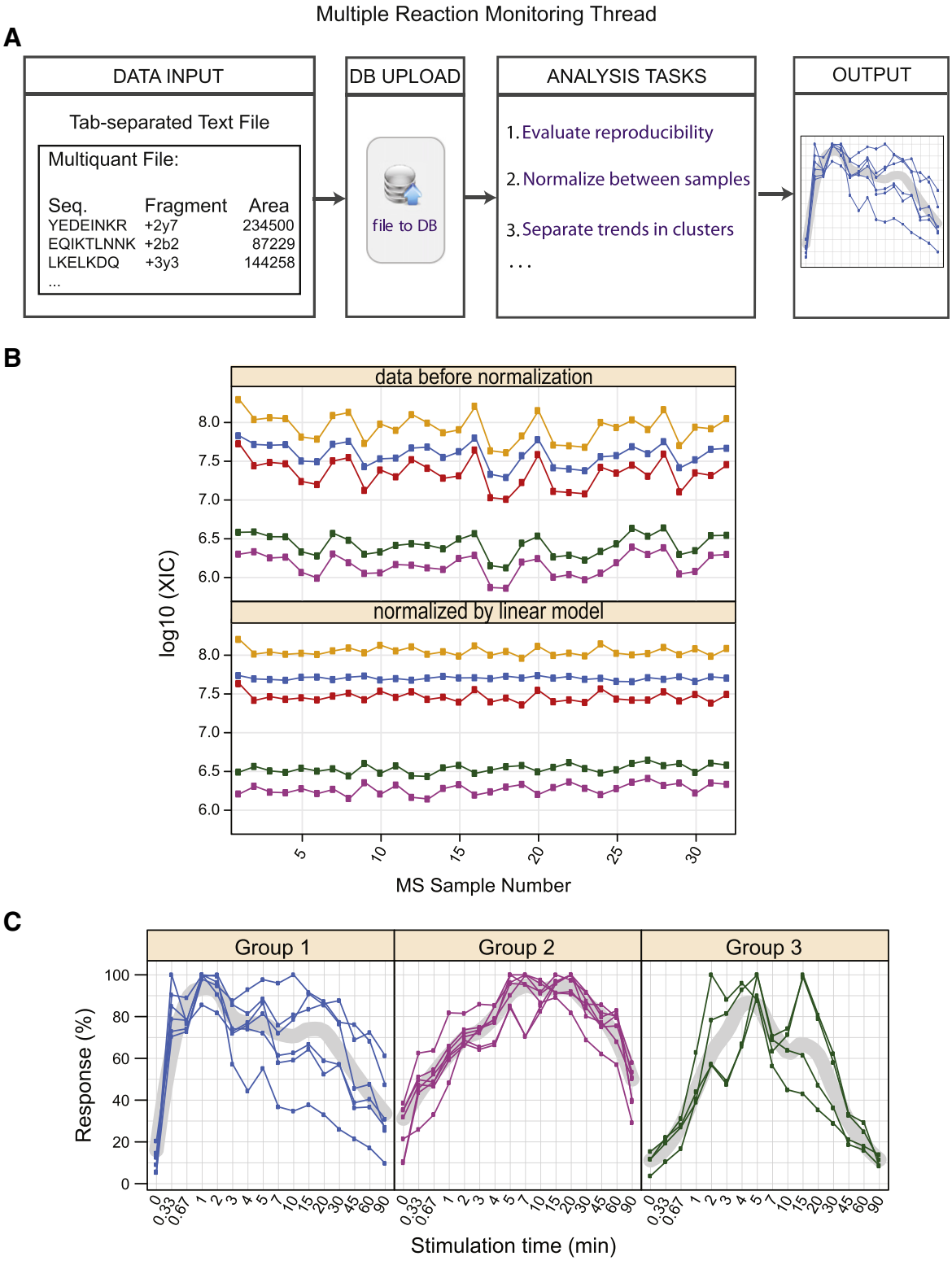


Fig. 1 – The CoreFlow pipeline. (A) CoreFlow is designed to integrate different types of experimental input data as well as computational scripts and database functionalities in order to correct and analyze data before publication-quality figures are produced. (B) From the main menu, a user can read CoreFlow's documentation, deploy the different database functionalities (create, browse or query database tables as well as import data from files using the 'file to DB function') or access analytical pipelines. Within the Analysis module, a project-specific pipeline is created as data inputs and analytic output expand with research progress.

on a local computer, an application server, or even on high-performance computing systems to improve speed. The work-load can be distributed across larger computational infrastructures through parallelization of R jobs and other scripts. Any size limits on data are imposed only by the operating environment (Linux, Apache, MySQL, PHP) and not by CoreFlow itself. Issues of browser timeouts are avoided by forking long tasks into child processes that maintain communication with the browser.

CoreFlow is now being released as an open-source application to assist other research scientists in managing complex data, correcting systematic errors, assessing the quality of



measurements, finding hidden patterns in experimental data, and ultimately, in modeling biological systems. The source code, database archive, documentation and demonstration are available at http://coreflow.mshri.on.ca, as is a link to a GitHub repository for code sharing and to a Google group for online discussions and assistance. The standard installation package and the on-line public version include a number of small and well-documented scripts (see Supplementary Table 1 for a list of scripts) outlining the workflow process. If desired, CoreFlow can be downloaded to run in a VirtualBox Linux environment.

We anticipate that CoreFlow will become a valuable resource for researchers who require a streamlined, flexible framework for analyzing and sharing biological data. Moving forward, we plan to release all code related to our publications in this format and we encourage other potential users to share their workflows too. By promoting the public release of code, CoreFlow can help eliminate the 'black box' issue [17–19] of unpublished code and reduce the use of unsuitable software [20] in publications. In this manner, we expect that the content of CoreFlow will become as valuable as the software itself.

Author contributions

A.P., T.P., R.L. & K.C. conceived the CoreFlow framework. R.L. & K.C. oversaw and managed the project. A.P. developed the code. A.P., P.C., E.M.S., R.L., & K.C. wrote the paper, and A.P., P.C., E.M.S., & K.C. wrote the Supplementary Information. E.M.S., Y.Z., R.D.V., J.S., & R.T. contributed experimental data and ideas for data analysis. M.O. tested CoreFlow. All authors commented on the manuscript.

Competing financial interests

The authors declare no competing financial interests.

Supplementary data to this article can be found online at http://dx.doi.org/10.1016/j.jprot.2014.01.023.

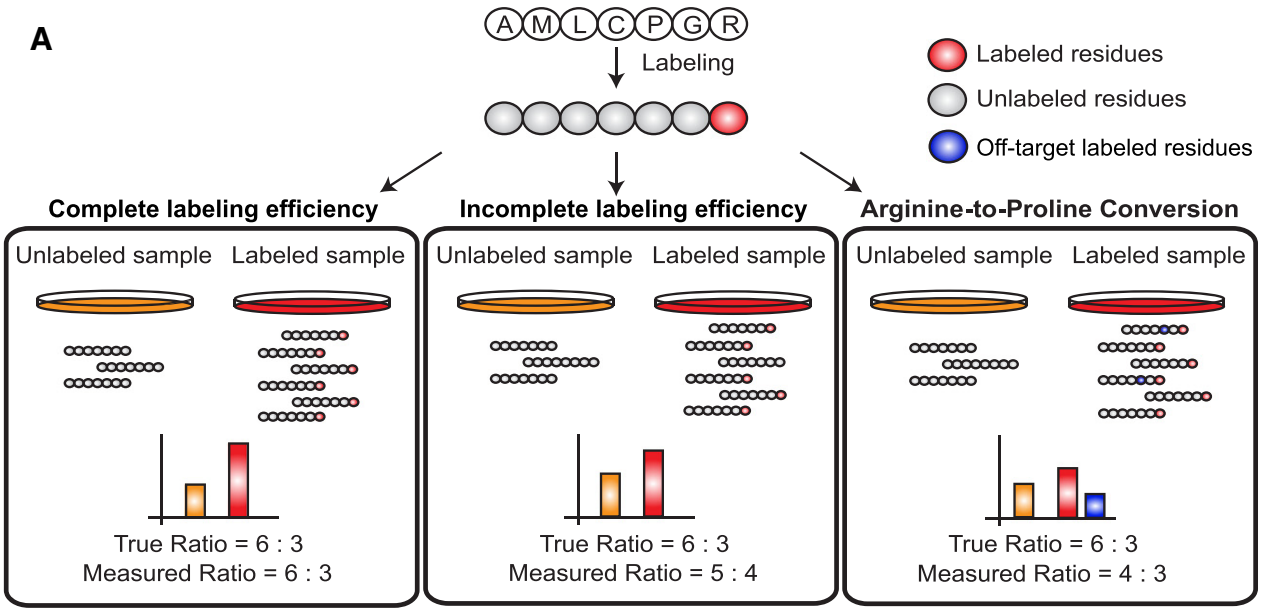
Acknowledgments

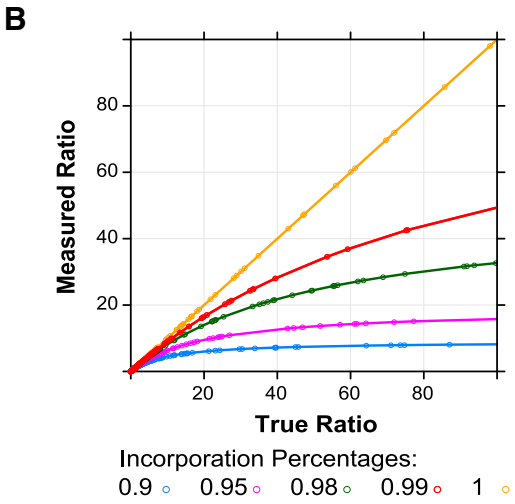
Thanks to Naveed Mohammed for systems administration support, and Søren Brunak, Jinho Kim and Evangelia Petsalakis for comments on the manuscript. This work was supported by funds to T.P. from Genome Canada through the Ontario Genomics Institute, the Ontario Research Fund GL^2 program, and the Canadian Institutes of Health Research (CIHR, MOP-6849, MOP-13466). R.L. is supported by the Lundbeck Foundation, the Human Frontier Science Program (HFSP), the Danish Council for Independent Research (FSS) and the European Research Council (ERC). R.D.V., J.S. and R.T. received fellowship support from CIHR.

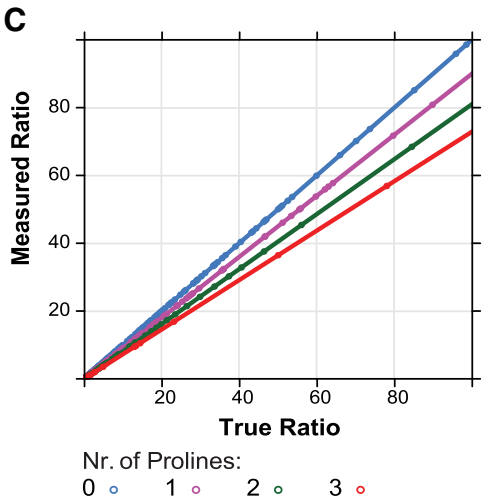
REFERENCES

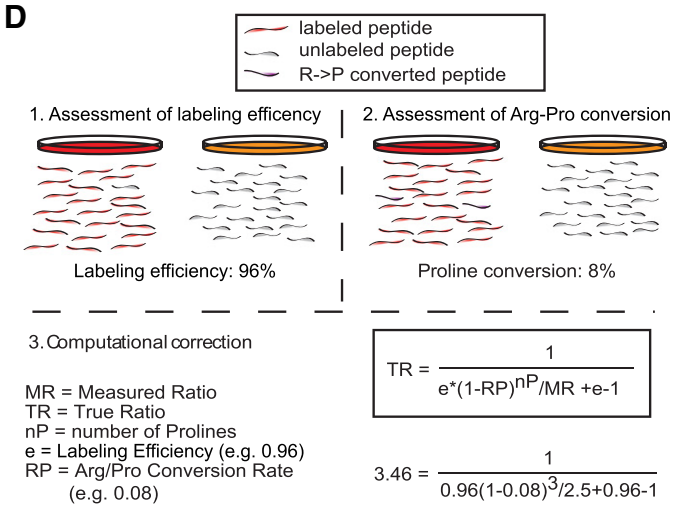
[1] Craig R, Beavis RC. TANDEM: matching proteins with tandem mass spectra. Bioinformatics 2004;20:1466–7.
[2] Yates III JR, Eng JK, McCormack AL, Schieltz D. Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. Anal Chem 1995;67:1426–36.
[3] Perkins DN, Pappin DJ, Creasy DM, Cottrell JS. Probability-based protein identification by searching sequence databases using mass spectrometry data. Electrophoresis 1999;20:3551–67.
[4] Cox J, Mann M. MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification. Nat Biotechnol 2008;26:1367–72.
[5] Liu G, Zhang J, Larsen B, Stark C, Breitkreutz A, Lin ZY, et al. ProHits: integrated software for mass spectrometry-based interaction proteomics. Nat Biotechnol 2010;28:1015–7.
[6] Craig R, Cortens JP, Beavis RC. Open source system for analyzing, validating, and storing protein identification data. J Proteome Res 2004;3:1234–42.
[7] Martens L, Hermjakob H, Jones P, Adamski M, Taylor C, States D, et al. PRIDE: the proteomics identifications database. Proteomics 2005;5:3537–45.
[8] Choi H, Larsen B, Lin ZY, Breitkreutz A, Mellacheruvu D, Fermin D, et al. SAINT: probabilistic scoring of affinity purification-mass spectrometry data. Nat Methods 2011;8:70–3.
[9] Nesvizhskii AI, Keller A, Kolker E, Aebersold R. A statistical model for identifying proteins by tandem mass spectrometry. Anal Chem 2003;75:4646–58.
[10] Chang CY, Picotti P, Huttenhain R, Heinzelmann-Schwarz V, Jovanovic M, Aebersold R, et al. Protein significance analysis in selected reaction monitoring (SRM) measurements. Mol Cell Proteomics 2011;11.
[11] Smoot ME, Ono K, Ruscheinski J, Wang PL, Ideker T. Cytoscape 2.8: new features for data integration and network visualization. Bioinformatics 2011;27:431–2.

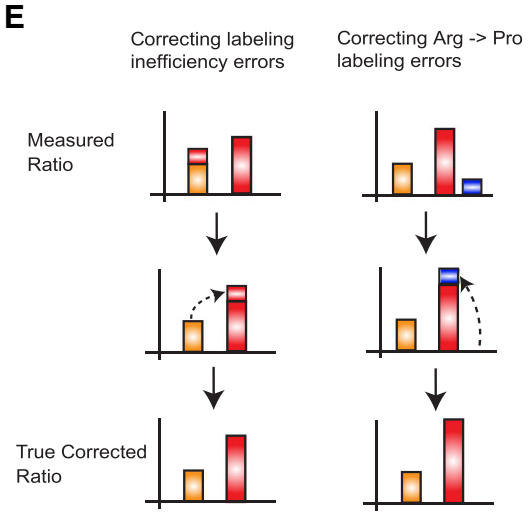
Fig. 2 – Multiple Reaction Monitoring (MRM) thread. (A) Example of an MRM thread in CoreFlow. Extracted ion chromatogram (XIC) values were loaded into a database using the 'file to DB' function. A new thread for MRM was created and populated with the indicated analysis tasks. Output from two of these tasks is shown in B and C. (B) Normalization of samples based on the signal of bait Shc1. Shc1 was immunoprecipitated from Rat-2 cells following a time course of epidermal growth factor (EGF) stimulation. For illustrative purposes, the samples are displayed by their sample number. Shc1 and interacting partners were quantified by MRM. A simple R linear model was used to compensate for variation between samples by adjusting Shc1 XIC levels to similar intensities. Transitions from Shc1 peptides are shown before (top panel) and after (bottom panel) normalization. Within a sample, all other proteins were then normalized to Shc1. (C) Temporal profiles of selected Shc1-interacting proteins. Once the data was normalized to Shc1, unsupervised clustering was applied to classify interacting partners of Shc1 according to their binding profiles. The clustering separated proteins into several groups of which three are shown. For this analysis, the maximal signal intensity for each protein during the time course was set at 100%. Group 1 proteins: PI3K-p110a, PI3K-p85a, PI3K-p85b, Lrrk1, FAM59A and ArhGEF5. Group 2 proteins: Dab2IP, Sgk269, PPP1ca, PPP1cg, ASAP1, ASAP2, Sgk223. Group 3 proteins: PTPN12, SHIP2, AP2s1, AP2a2. Trend smoothing was applied by using the spline function in R to identify the average trend (gray shaded line). Complete details of the Shc1 study are provided in [16].


A
A M L C P G R
Labeling
Labeled residues
Unlabeled residues
Off-target labeled residues
Complete labeling efficiency
Incomplete labeling efficiency
Arginine-to-Proline Conversion
Unlabeled sample
Labeled sample
True Ratio = 6 : 3
Measured Ratio = 6 : 3
True Ratio = 6 : 3
Measured Ratio = 5 : 4
True Ratio = 6 : 3
Measured Ratio = 4 : 3


B
Measured Ratio
True Ratio
20
40
60
80
Incorporation Percentages:
0.9
0.95
0.98
0.99
1


C
Measured Ratio
True Ratio
20
40
60
80
Nr. of Prolines:
0
1
2
3


D
labeled peptide
unlabeled peptide
R->P converted peptide
1. Assessment of labeling efficency
2. Assessment of Arg-Pro conversion
Labeling efficiency: 96%
Proline conversion: 8%
3. Computational correction
MR = Measured Ratio
TR = True Ratio
nP = number of Prolines
e = Labeling Efficiency (e.g. 0.96)
RP = Arg/Pro Conversion Rate
(e.g. 0.08)
TR = 1 / e*(1-RP)^nP/MR +e-1
3.46 = 1 / 0.96(1-0.08)^3/2.5+0.96-1


E
Correcting labeling inefficiency errors
Correcting Arg -> Pro labeling errors
Measured Ratio
True Corrected Ratio

[12] Saez-Rodriguez J, Goldsipe A, Muhlich J, Alexopoulos LG, Millard B, Lauffenburger DA, et al. Flexible informatics for linking experimental data to mathematical models via DataRail. Bioinformatics 2008;24:840–7.
[13] Vanderlaan RD, Hardy WR, Kabir MG, Pasculescu A, Jones N, Detombe PP, et al. The ShcA phosphotyrosine docking protein uses distinct mechanisms to regulate myocyte and global heart function. Circ Res 2011;108:184–93.
[14] Jorgensen C, Sherman A, Chen GI, Pasculescu A, Poliakov A, Hsiung M, et al. Cell-specific information processing in segregating populations of Eph receptor ephrin-expressing cells. Science 2009;326:1502–9.
[15] Tan CS, Pasculescu A, Lim WA, Pawson T, Bader GD, Linding R. Positive selection of tyrosine loss in metazoan evolution. Science 2009;325:1686–8.
[16] Zheng Y, Zhang C, Croucher DR, Soliman MA, St-Denis N, Pasculescu A, et al. Temporal regulation of EGF signaling networks by the scaffold protein Shc1. Nature 2013;499:166–71.
[17] Morin A, Urban J, Adams PD, Foster I, Sali A, Baker D, et al. Research priorities. Shining light into black boxes. Science 2012;336:159–60.
[18] Mesirov JP. Computer science. Accessible reproducible research. Science 2010;327:415–6.
[19] Ince DC, Hatton L, Graham-Cumming J. The case for open computer programs. Nature 2012;482:485–8.
[20] Martin SF, Falkenberg H, Dyrlund TF, Khoudoli GA, Mageean CJ, Linding R. PROTEINCHALLENGE: crowd sourcing in proteomics analysis and software development. J Proteomics 2013;88:41–6.

Fig. 3 – Computational correction of SILAC inaccuracies using CoreFlow. (A) Incomplete labeling (center) and arginine-to-proline conversion (right) are two phenomena that can make SILAC ratios inaccurate and far from their true ideal values (left). (B) Incomplete labeling deviates the measured ratio from the true ratio in a manner that is non-linearly correlated with the SILAC incorporation percentage. For severely unlabeled samples (e.g. 0.9 incorporation), it becomes impossible to measure ratios that are above a threshold. (C) Arginine-to-proline conversion is, unsurprisingly, dependent on the number of prolines in the MS peptide; peptides with more prolines are more prone to wrong measurements. (D) After assessing labeling deficiency and arginine-to-proline conversion rates in the sample of interest (please refer to the Supplementary Text for further details), the measured ratios can be corrected using the formula provided. (E) By using this formula, measured SILAC ratios affected by both labeling and arginine-to-proline conversion inaccuracies can be converted to correct ratios that match unaffected samples.